1017 U.S. PTO
10/083723
02/25/02

```
///////////////////////////////////////////////////////////////////////////
//
///////
//
//   SYSTEM:           Polar HRMCom
//
//   UNIT FILE NAME:   HRMCOM.H
//
//   MODULE:                       HRMCOM.DLL
//
//   AUTHORS:          MEr / Polar Electro Oy
//
//   VERSION:          1.2
//
//   DATE:             20.03.2001
//
//   ABSTRACT:         Main header file for Polar HRMCOM.DLL function
library file.
//
//   REMARKS:
//
//   COPYRIGHT (C) 2001 BY POLAR ELECTRO OY
//
///////////////////////////////////////////////////////////////////////////
//
///////

□

///////////////////////////////////////////////////////////////////////////
//
///////
//
//       BOOLEAN VARIABLES
//       -----------------
//
//       Function library uses boolean variables as following:
//
//               TRUE    equals  1
//               FALSE   equals  0
//
///////////////////////////////////////////////////////////////////////////
//
///////

///////////////////////////////////////////////////////////////////////////
//
///////
```

```
//
//      FUNCTION CALLS
//      --------------
//
//      Definition of function calls:
//
//              __declspec (dllexport) BOOL CALLBACK fnHRMCom...
//
//      can be replaced by
//
//              BOOL fnHRMCom...
//
///////////////////////////////////////////////////////////////////////
//
///////

///////////////////////////////////////////////////////////////////////
//
///////
//
//      INITIALIZING DATA STRUCTURES
//      ----------------------------
//
//      It is recommended to initialize all data structure always befo
re
usage by using
//      for example the functions memset or ZeroMemory. Uninitialized
data structure
//      passed to functions may cause errors in communication.
//
///////////////////////////////////////////////////////////////////////
//
///////

///////////////////////////////////////////////////////////////////////
//
///////
//
//      POLAR S-SERIES MONITOR FEATURES
//      -------------------------------
//
//      Feature                              S210    S410    S510    S610
S710    S810    E200    E600
//
-----------------------------------------------------------------------
--
--------
//      Watch Settings          x               x               x
```

```
x                x                x               x4               x4
//      Exercise Sets             x                x                x

x                x                x1              x4               x4
//      User Settings             x                x                x

x2               x2               x2              x4               x4
//      Reminders

x                x                x
//      Exercise Profiles

x
//      Monitor Bitmaps

x                x                x
//      Bikes

x                                 x3
//
//      x       = feature available
//      x1      = Only one exercise set available
//      x2      = User settings extended with user name string
//      x3      = Also power output settings
//      x4      = Education models' features are limited, see function

definitions
//
//      For more details about feature difference, refer to each
function call definitions
//      and HR monitor user's manuals. Another good hint is also to us
e
Polar Precision
//      Performance SW 3.0 and it's HR Monitor Connection. This softwa
re
products utilizes
//      HRMCom.dll function library.
//
////////////////////////////////////////////////////////////////////////////
//
///////

////////////////////////////////////////////////////////////////////////////
//
///////
//
// POLAR CHARACTER SET
// -------------------
//
//       The following characters are valid at text strings in settings
:
//       - Capital letters:      ABCDEFGHIJKLMNOPQRSTUVWXYZ
//       - Small letters:        abcdefghijklmnopqrstuvwxyz
//       - Numbers:                      0123456789
//       - Special chars:        -%/()*+.:? and space
//
//       Unrecognized characters will be converted automatically to
spaces.
//       The text strings have to be ended by zero character (NULL).
//       Strings can be checked before sending by using function
fnHRMCom_CheckPolarCharString
//
__declspec (dllexport) BOOL CALLBACK fnHRMCom_CheckPolarCharString
(LPTSTR);
//
//////////////////////////////////////////////////////////////////////////
//
///////

//////////////////////////////////////////////////////////////////////////
//
///////
//
//       DATE FORMAT
//       -----------
//
//       Date values are processed in yyyymmdd format:
//
//               yyyy    year    4 digits
//               mm              month   2 digits
//               dd              day             2 digits
//
//       For example:    August 2nd 2000         => 20000802
//                                       December 24th 2003      =>
20031224
//
//       NOTE: Leading zero with days and months is always obligatory.
//
//////////////////////////////////////////////////////////////////////////
//
///////

//////////////////////////////////////////////////////////////////////////
//
///////
```

```
//
//      ERROR CHECKING
//      --------------
//
//      All input parameters will be checked before sending to heart
rate monitor.
//      If any erratic values are determined, function call returns
FALSE and does not
//      continue sending data to monitor. The latest error code can be

checked by function:
//      iError = fnHRMCom_GetErrorCode ();
//
__declspec (dllexport) int CALLBACK fnHRMCom_GetErrorCode

(void);
//
//      ## UNDER CONSTRUCTION ## //
//
//////////////////////////////////////////////////////////////////////
//
///////

//////////////////////////////////////////////////////////////////////
//
///////
//
//      POLAR HR MONITOR TYPES
//      ----------------------
//
#define         HRM_S210
8                       // Polar S210
#define         HRM_S410
9                       // Polar S410
#define         HRM_S510
10                      // Polar S510
#define         HRM_S610
11                      // Polar S610
#define         HRM_S710
12                      // Polar S710
#define         HRM_S810
13                      // Polar S810
#define         HRM_E200
14                      // Polar E200 Education HRM
#define         HRM_E600
15                      // Polar E600 Education HRM
//
//      NOTE: Education HR monitors are regionally available.
```

```
//
////////////////////////////////////////////////////////////////////////////////////
//
///////

////////////////////////////////////////////////////////////////////////////////////
//
///////
//
//      MONITOR CONNECTION METHODS
//      --------------------------
//
#define         HRMCOM_CONNECTION_UPLINK                0
#define         HRMCOM_CONNECTION_IR                    1
//
//      Polar UpLink technology can be used only for transferring
settings from computer
//      to Polar S-series HR monitor (one-way). "Read" functions can b
e
called with
//      HRMCOM_CONNECTION_UPLINK as connection method, but method is
automatically
//      changed to HRMCOM_CONNECTION_IR. Infrared conenction is
automatically two-way,
//      this means all the settings etc. can be read and written.
//
//      When infrared is used for writing or reading data to/from HR
monitor, the communication
//      have to be started by using function
fnHRMCom_StartIRCommunication. After calling this
//      function, all the other reading and writing functions can be
used normally. To end
//      infrared communication, call function
fnHRMCom_EndIRCommunication.
//
////////////////////////////////////////////////////////////////////////////////////
//
///////

////////////////////////////////////////////////////////////////////////////////////
//
///////
//
//      POLAR UPLINK WAVE FILES
//
//      By default wave file (random name to Temp folder) will be
created, played and deleted.
//      Wave file will be created automatically to Temp folder defined
```

```
at the Windows system.
//      The new wave file will be automatically named as HRMxxx.WAV,
where xxx is a random number.
//      The playing of wav file do not allow cancelling.
//
///////////////////////////////////////////////////////////////////////
//
///////
□

////////////////////////
////////////////////////
//
//      LIBRARY VERSION DATA
//
////////////////////////
////////////////////////

// Get hrmcom library file version
__declspec (dllexport) int CALLBACK fnHRMCom_GetLibraryVersion
(void);

// Version 1.00 will be returned as 100

////////////////////////
////////////////////////
//
//      GENERAL SETTINGS DATA
//
////////////////////////
////////////////////////

//      The following data structure will be used with the most of the

functions to give general information
//      about communication, for example are we using Polar UpLink or
Infrared connection.

typedef struct
{
        int             iSize;                          // Structure
size for version control
                                                                // Get

using sizeof (STRUCTURE)

        int             iConnection;            // Connection method:
```

```
HRMCOM_CONNECTION_UPLINK or HRMCOM_CONNECTION_IR
                                                                        // NOT
E:
Polar UpLink connection can be used only for writing information to HR

monitor:

        int             iMonitorID;                               // Unique
monitor ID, 0 = message to all monitors
                                                                        //
Monitor will accept the messages if monitor id to send is same as
already
                                                                        // set

by User settings or if message was meant for all monitors available.
                                                                        // Oth
er
ID numbers used mainly with IR communication

        TCHAR   szWaveFile[MAX_PATH];// Wave file name, use NULL to
create random file name
                                                                        // to

Temp folder (MAX_PATH = 260)

                                                                        //
EXCLUSION FLAGS
                                                                        //
---------------
        BOOL    bNoCreateWave;          // Don't create wave file at
all, this allows testing of values in the data structure
        BOOL    bNoPlayWave;            // Don't play created Polar
UpLink WAV file
        BOOL    bNoDeleteWave;          // Don't delete created Polar
UpLink WAV file after it have been played

                                                                        // DAT
A
FILE MANAGEMENT
                                                                        //
--------------------
        BOOL    bLoadFromDataFile;      // Load information from binar
y
data file, file name have to be at szWaveFile
                                                                        // If
trying to load the data file with not the same data as data structure
                                                                        //
specified in call, all the calling functions will return FALSE
                                                                        // Whe
```

```
n
file will be loaded, other actions (create, play, delete wave) are not

done.
                                                                     // If
loaded file includes incorrect data, default values will be set
automatically.

        BOOL    bSaveAsDataFile;        // Save information to binary
data file, file name have to be at szWaveFile
                                                                     // Whe
n
file will be saved, other actions (create, play, delete wave) are not
done.

                                                                     //
CONNECTION DIALOG
                                                                     //
-----------------
        BOOL    bConnectionDlg;         // Usage of connection dialog
to
user
        HWND    hOwnerWnd;                      // Owner window handle

to connection dialog
                                                                     // If
connection dialog has been selected to be shown, owner window
                                                                     //
handle have to be specified. If not, dialog won't be shown and
connection fails.
                                                                     // If
connection dialog is not in use, this parameter is ingnored.

        TCHAR   szDlgMsg[50];           // Connection dialog message t
o
user, max 50 characters
                                                                     // If
message text is not specified, default English texts will be used
                                                                     // If
connection dialog is not in use, this parameter is ingnored.

                                                                     // MIS
C
PARAMETERS
                                                                     //
---------------
        BOOL    bFixErrors;                     // Errors in settings
can be fixed automatically and error messages
```

```
                                                                        // are
not returned in normal cases.

        int             iParam;                                      // Parameter
reserved for future usage, use zero
        long    lParam;                                      // Parameter reserved
for future usage, use zero

} POLAR_SSET_GENERAL;



////////////////////////
////////////////////////
//
//      WATCH SETTINGS
//
////////////////////////
////////////////////////

//      All Polar S-series HR monitors do have two independent time
zones. The active time zone
//      can be selected with iActiveTime.

typedef struct
{
        int             iSize;                                       // Structure
size for version control
                                                                        // Get
using sizeof (STRUCTURE)

        int             iTime1;                                      // Time in
seconds from midnight (0:00:00), max 23:59:59 = 86399 sec
                                                                        // If
iTime1 = -1, current system time is automatically set to iTime1
        int             iTime2;                                      // Time in
seconds from midnight (0:00:00), max 23:59:59 = 86399 sec
                                                                        // If
iTime2 = -1, current system time is automatically set to iTime2
                                                                        // Onl
y
full hours and minutes are valid, seconds will be set to zero

        int             iTime1HourMode;              // 0 = 24h mode, 1 = 1
2h
mode
```

```
        int                iTime2HourMode;          // 0 = 24h mode, 1 = 1
2h
mode
        int                iActiveTime;             // 0 = time1 active, 1
 =
time2 active
        int                iDate;                           // Date in
format yyyymmdd, Jan 1 2000 - Dec 31 2099
                                                                    // If
iDate = -1, current system date is automatically set to iDate
        BOOL    bAlarmEnabled;              // FALSE = off, TRUE = on
        int                iAlarmTime;                      // Time in
seconds from midnight (0:00:00), max 23:59:59 = 86399 sec
                                                                    // Onl
y
full hours and minutes are valid, seconds will be set to zero

} POLAR_SSET_WATCH;

__declspec (dllexport) void CALLBACK fnHRMCom_ResetWatchSettings
(POLAR_SSET_WATCH*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendWatchSettings
(POLAR_SSET_GENERAL*, POLAR_SSET_WATCH*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_ReadWatchSettings
(POLAR_SSET_GENERAL*, POLAR_SSET_WATCH*);

// NOTE: Education HR monitors E200 and E600 do have only one time
(iTime1) and no alarm available.
// Set iTime2 to the same as iTime1, hour mode should be the same for
both times.
// Alarm time should be zero and alarm should be not enabled.

□

////////////////////////
////////////////////////
//
// EXERCISE SET
//
////////////////////////
////////////////////////

// Exercise Set information will be send to monitor one set at a time.
// Exercise Set can be set as an active set to monitor (i.e. set will
be
shown
// as the first set when next time starting exercise).
```

```
typedef struct
{
        int             iSize;                          // Structure
size for version control
                                                                // Get

using sizeof (STRUCTURE)

        int             iSetNumber;                     // Exercise se
t
number: 0, 1, 2, 3, 4, 5.
                                                                // Set

number 0 can be used only for setting "Basic Set" active.
                                                                // Set
s
2 - 5 are not available for all monitors (see Polar S-series Monitor
Features).

        BOOL    bActiveSet;                     // Will this set to be

set as an active set in monitor? TRUE/FALSE
        TCHAR   szName[8];                      // Exercise set name
(see Polar Character Set)
                                                                //
String can be checked using function fnHRMCom_CheckPolarCharString
                                                                // Max

number of characters is 7 + ending zero
                                                                //
"BasicUse" name is permanent for S610, S710 and S810, it can't be
modified.

        // Timers
        BOOL    bTimer1Enabled;         // Timer 1 enabled TRUE/FALSE
        int             iTimer1;                        // Timer 1 val
ue
in seconds, max 99 min 59 sec (= 5999 sec)
        BOOL    bTimer2Enabled;         // Timer 2 enabled TRUE/FALSE
        int             iTimer2;                        // Timer 2 val
ue
in seconds, max 99 min 59 sec
                                                                // Tim
er
2 used as interval timer, if intervals enabled.
        BOOL    bTimer3Enabled;         // Timer 3 enabled TRUE/FALSE
        int             iTimer3;                        // Timer 3 val
ue
```

```
in seconds, max 99 min 59 sec

        // HR Limits
        BOOL    bHRLimit1Enabled;       // HR Limits 1 enabled
        int             iHRLimit1Upper;         // HR Limit 1 upper
value 30 - 240 bpm
        int             iHRLimit1Lower;         // HR Limit 1 lower
value 30 - 240 bpm (must be less than upper limit)
        BOOL    bHRLimit2Enabled;       // HR Limits 2 enabled
        int             iHRLimit2Upper;         // HR Limit 2 upper
value 30 - 240 bpm
        int             iHRLimit2Lower;         // HR Limit 2 lower
value 30 - 240 bpm (must be less than upper limit)
        BOOL    bHRLimit3Enabled;       // HR Limits 3 enabled
        int             iHRLimit3Upper;         // HR Limit 3 upper
value 30 - 240 bpm
        int             iHRLimit3Lower;         // HR Limit 3 lower
value 30 - 240 bpm (must be less than upper limit)

        BOOL    bMaxHRInUse;            // Are HR limit values in
percentage of maximum HR given in iMaxHR variable?
                                                                // If
TRUE, all HR limit values are used as percentage values (50 - 100%)
        int             iMaxHR;                         // Maximum HR
value to be used for calculation of HR limit values.
                                                                // HR
value in bpm, 100 - 240 bpm

        // Intervals
        BOOL    bIntervalsEnabled;      // TRUE/FALSE
        int             iIntervalType;          // 0 = manual, 1 = tim
er
(use Timer2), 2 = HR,
                                                                // 3 =

distance (distance only with cycling models)
        int             iIntervalCount;         // The number of
intervals, 0 - 30 (0 = unlimited)
        int             iIntervalEndHR;         // Interval ending HR
bpm 10 - 240 bpm
        int             iIntervalDistKm;        // The distance of
interval in 0.1 km (max 99.9 km)
                                                                // If
monitor does not support cycling features, this value is ignored
        int             iIntervalDistMiles;     // The distance of
interval in 0.1 miles (max 99.9 miles)
                                                                // If
monitor does not support cycling features, this value is ignored
```

```
                                                                        // If
both distance values are specified, km value takes precedence.

        // Recovery
        BOOL    bRecoveryEnabled;       // TRUE/FALSE
        int             iRecoveryType;          // 0 = timer recovery,
 1
= HR recovery
                                                                        // 2 =

distance recovery  (distance only with cycling models)
        int             iRecoveryTime;          // Recovery time in
seconds, max 99 min 59 sec (max 5999 sec)
        int             iRecoveryHR;            // recovery HR value 1
0
- 240 bpm
        int             iRecoveryDistKm;        // The distance of
recovery in 0.1 km (max 99.9 km)
                                                                        // If
monitor does not support cycling features, this value is ignored
        int             iRecoveryDistMiles;     // The distance of
recovery in 0.1 miles (max 99.9 miles)
                                                                        // If
monitor does not support cycling features, this value is ignored
                                                                        // If
both distance values are specified, km value takes precedence.

} POLAR_SSET_EXERCISESET;

__declspec (dllexport) void CALLBACK fnHRMCom_ResetExerciseSet
(int, POLAR_SSET_EXERCISESET*, int, int);       // ..., iExerciseType,

iMonitor
__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendExerciseSet
(POLAR_SSET_GENERAL*, POLAR_SSET_EXERCISESET*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_ReadExerciseSet
(POLAR_SSET_GENERAL*, int, POLAR_SSET_EXERCISESET*);

//      Integer value at resetting and reading functions include exe s
et
number (1 - 5)
//      If sending was succesfull, function returns TRUE, otherwise
FALSE

// NOTE: Education HR monitors E200 and E600 do have only one Exercise

Set, iSetNumber should be 1 (one).
// E200 and E600 HR monitors do have the following features:
```

```
//      - Timers 1 and 2
//      - HR Limits 1
//      - Recovery calculation (type timer always)
// Other exercise settings should be set to default values
□

/////////////////////////
/////////////////////////
//
// USER SETTINGS
//
/////////////////////////
/////////////////////////

//      User settings include both information about the person and th
e
usage of monitor features.
//      All the settings are not available in all Polar S-serie monito
r,
see more details
//      from monitor specifications. If any data send to monitor is no
t
supported, it will
//      be ignored automatically.

typedef struct
{
        int             iSize;                          // Structure
size for version control
                                                                // Get

using sizeof (STRUCTURE)
        // Information about user
        int             iDateOfBirth;           // Date of birth in
format yyyymmdd, Jan 1 1921 - Dec 31 2020
        int             iActivityLevel;         // Activity level: 0 =

low, 1 = moderate, 2 = high, 3 = top
        int             iMaxHR;                         // Maximum hea
rt
rate value 100 - 240 bpm
        int             iVO2max;                        // VO2max valu
e
10 - 95 mmol/l/kg
        int             iUserSex;                       // Sex of user
:
0 = male, 1 = female
```

```
        int             iWeightKg;                      // Weight in
kilograms: 0, 20 - 199 kg
        int             iWeightLbs;                     // Weight in
pounds: 0, 44 - 499 lbs
                                                                // If
both weight values are specified, kg value takes precedence.
        int             iHeightCm;                      // Height in
centimeters, 0, 90 - 211 cm
        int             iHeightFt;                      // Height in
feet: 0, 3 - 7 ft
        int             iHeightInches;          // Height in inches: 0
 -
11 inches
                                                                // If
both height values are specified, cm value takes precedence.

        TCHAR   szName[8];                      // User name (see Pola
r
Character Set)
                                                                //
String can be checked using function fnHRMCom_CheckPolarCharString
                                                                // Max
number of characters is 7 + ending zero
                                                                // If
monitor does not support user name, this value is ignored

        // Monitor Features and Functions
        int             iMonitorID;                     // Monitor ID
number (for example player number) 0 - 99
        BOOL    bOwnCal;                        // OwnCal calculation
enabled TRUE/FALSE
        BOOL    bHRMaxP;                        // HRmax-p calculation
enabled TRUE/FALSE
        BOOL    bOwnIndex;                      // OwnIndex calculatio
n
enabled TRUE/FALSE
        BOOL    bAltimeter;                     // Altimeter enabled
TRUE/FALSE, available only for S710
        BOOL    bButtonSound;           // Button sounds enabled
TRUE/FALSE
        BOOL    bOptionsLock;           // Options mode lock enabled
TRUE/FALSE
        BOOL    bHelp;                          // Feature help functi
on
enabled TRUE/FALSE
        BOOL    bUS_Units;                      // Measurement units:
```

```
FALSE = EURO units, TRUE = US units
        int             iSamplingRate;          // 0 = 5s, 1 = 15s, 2
=
60s, 3 = R-R intervals
                                                                    //
Sampling rate selection is available only with S610, S710 and S810
                                                                    // R-R

intervals recording is available only with S810
                                                                    //
Monitor S210 do not have sampling rate selection
                                                                    //
Monitors S410 and S510 have always dynamic sampling rate
        int             iHeartTouch;            // Usage of Wireless
Button trigger (heart touch feature)
                                                                    // 0 =

normal, 1 = lap, 2 = change display and limits
                                                                    //
Wireless button action selection is available with S610, S710 and S810
        int             iRLXBaseLine;           // Relaxation base lin
e
only for S810, 4 - 150 mseconds
        BOOL    bOnlineRecording;       // Online recording enabled
TRUE/FALSE, S810 only

} POLAR_SSET_USER;

__declspec (dllexport) void CALLBACK fnHRMCom_ResetUserSettings
(POLAR_SSET_USER *);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendUserSettings
(POLAR_SSET_GENERAL*, POLAR_SSET_USER*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_ReadUserSettings
(POLAR_SSET_GENERAL*, POLAR_SSET_USER*);

//      If sending was succesfull, function returns TRUE, otherwise
FALSE

// NOTE: Education HR monitors E200 and E600 do have only the followin
g
features:
// Options Lock, User Name, Monitor ID, Sampling Rate (E600 only)

//////////////////////////////////////////////////////////////////////
//
///////
//
// VO2max and HRmax-p values are used in OwnCal calories calculation
// and those values can be updated as follows:
//
//      PC using UpLink/IR ----->
//      OwnIndex from FitTest --> UserSet in Monitor ---> OwnCal
calculation in monitor
//      Manually set ----------->
//
/////////////////////////////////////////////////////////////////////////
//
///////

□

/////////////////////////
/////////////////////////
//
// REMINDER
//
/////////////////////////
/////////////////////////

//      Reminders are available with S610, S710 and S810 heart rate
monitors. There are
//      7 reminder "slots" available in each HR monitor and those can
be
modified only by using
//      computer. Each reminder can be individually set to be activate
d
at selected date & time.
//      One reminder at time can be sent to HR monitor, select reminde
r
"slot" to be updated by iNumber.
//      Reminder can be repeated automatically hourly, daily, monthly,

weekly, monthly and yearly.
//      An exercise (ExeSet / ExeProfile) can be set to be active afte
r
reminder has alarmed.

typedef struct
{
        int             iSize;                          // Structure
size for version control
                                                                // Get

using sizeof (STRUCTURE)

        int             iNumber;                        // Number of
reminder, 0 - 6
        BOOL    bActive;                        // Reminder activated
TRUE/FALSE
        int             iDate;                          // Date of
reminder in format yyyymmdd, Jan 1 2000 - Dec 31 2020
        int             iTime;                          // Time in
seconds from midnight (0:00:00), max 23:59:59 = 86399 sec
                                                                // Onl
y
full hours and minutes are valid, seconds will be set to zero
        int             iRepeat;                        // Repetition
of
reminder. 0 = Off,  1 = Hourly,
                                                                // 2 =

Daily, 3 = Weekly,  4 = Monthly, 5 = Yearly
        int             iExercise;                      // S810:
Exercise Profile to be set as default profile after reminder alarm
                                                                // 0 =

Off,  1 = BasicUse, 2 - 8 Profile Number (remember to update also
exercise profiles)
                                                                // S61
0
& S710: Exercise Set to be set as default profile after reminder alarm
                                                                // 0 =

Off,  1 = BasicUse, 2 - 7 ExeSet Number (remember to update also
exercise sets)
        TCHAR   szText[8];                      // Reminder Text (see
Polar Character Set)
                                                                //
String can be checked using function fnHRMCom_CheckPolarCharString
                                                                // Max

number of characters is 7 + ending zero

} POLAR_SSET_REMINDER;

__declspec (dllexport) void CALLBACK fnHRMCom_ResetReminder

(int, POLAR_SSET_REMINDER*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendReminder

(POLAR_SSET_GENERAL*, POLAR_SSET_REMINDER*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_ReadReminder

(POLAR_SSET_GENERAL*, int, POLAR_SSET_REMINDER*);
```

```
//      Integer value at resetting and reading functions include
reminder number (0 - 6)
//      If sending was succesfull, function returns TRUE, otherwise
FALSE

□

/////////////////////////
/////////////////////////
//
// BIKES
//
/////////////////////////
/////////////////////////

//      Bike information is available only with S510 and S710. Bike's
power settings will
//      be transferred only to Polar S710 HR monitor.

typedef struct
{
        // Bike Information
        TCHAR   szBikeID[5];            // Bike ID (name) (see Polar
Character Set)
                                                                       //
String can be checked using function fnHRMCom_CheckPolarCharString
                                                                       // Max

number of characters is 4 + ending zero
        int             iWheelSize;                     // Wheel size
in
millimeters (1000 - 3000 mm)
        BOOL    bAutoStart;                     // Is autostart featur
e
in use TRUE/FALSE
        BOOL    bSensorSpeed;           // Speed sensor in use
TRUE/FALSE. This flag is not in use, speed sensor is always in use.
        BOOL    bSensorCadence;         // Cadence sensor in use
TRUE/FALSE
        BOOL    bSensorPower;           // Power sensor in use
TRUE/FALSE
                                                                       // Pow
er
sensor is available only with Polar S710 HR monitor
                                                                       // If
monitor does not support power sensor, this value is ignored
```

```
        // Power Sensor Settings
        int             iChainMass;                     // Weight of
chain in grams (200 - 400 g)
        int             iChainLength;           // Length of chain in
mm
(1000 - 2000 mm)
        int             iChainWank;                     // The length
of
vibrating part (span) chain in mm (300 - 600 mm)

} POLAR_BIKE_INFO;

typedef struct
{
        int             iSize;                          // Structure
size for version control
                                                                // Get

using sizeof (STRUCTURE)

        int             iBikeInUse;                     // Which bike
has been selected to be in use right now?
                                                                // 0 =

Bike1, 1 = Bike2, 2 = None (no cycling features in use)
                                                                // If
Bike1 is in use, some cycling sensor (speed, cadence or
                                                                //
power) have to be in use.
        POLAR_BIKE_INFO Bike[2];

} POLAR_SSET_BIKES;

__declspec (dllexport) void CALLBACK fnHRMCom_ResetBikeSettings
(POLAR_SSET_BIKES*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendBikesSettings
(POLAR_SSET_GENERAL*, POLAR_SSET_BIKES*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_ReadBikesSettings
(POLAR_SSET_GENERAL*, POLAR_SSET_BIKES*);

//      If sending was succesfull, function returns TRUE, otherwise
FALSE

□

////////////////////////
////////////////////////
//
```

```
// EXERCISE PROFILE
//
/////////////////////////
/////////////////////////

typedef struct
{
        BOOL    bPhaseEnabled;          // Has exercise phase been
enabled? TRUE/FALSE

        // HR Limits
        BOOL    bHRLimitEnabled;        // HR Limits enabled
        int             iHRLimitUpper;          // HR Limit upper valu
e
30 - 240 bpm
        int             iHRLimitLower;          // HR Limit lower valu
e
30 - 240 bpm (must be less than upper limit)

        // Interval period
        BOOL    bIntervalsEnabled;      // Is entire work period
enabled? TRUE/FALSE
        int             iIntervalType;          // 0 = manual, 1 =
timer, 2 = End HR
        int             iIntervalCount;         // The number of
intervals, 0 - 30 (0 = unlimited)
        int             iIntervalTimer;         // Timer value in
seconds, max 99 min 59 sec (= 5999 sec)
        int             iIntervalEndHR;         // Interval ending HR
bpm 10 - 240 bpm

        // Recovery period
        BOOL    bRecoveryEnabled;       // Is entire recovery period
enabled ? TRUE/FALSE
        int             iRecoveryType;          // 0 = timer recovery,
 1
= HR recovery
        int             iRecoveryTime;          // Recovery time in
seconds, max 99 min 59 sec (max 5999 sec)
        int             iRecoveryHR;            // recovery HR value 1
0
- 240 bpm

} POLAR_EXEPHASE;

typedef struct
{
        int             iSize;                          // Structure
size for version control
                                                                  // Get

using sizeof (STRUCTURE)

        int             iNumber;                        // Exercise
profile number, 1 - 7
        BOOL    bActiveProfile;         // Will this profile to be set

as an active set in monitor? TRUE/FALSE
        TCHAR   szName[8];                      // Exercise profile na
me
(see Polar Character Set)
                                                                  //
String can be checked using function fnHRMCom_CheckPolarCharString
                                                                  // Max

number of characters is 7 + ending zero

        BOOL    bMaxHRInUse;            // Are HR limit values in
percentage of maximum HR given in iMaxHR variable?
                                                                  // If
TRUE, all HR limit values are used as percentage values (50 - 100%)
        int             iMaxHR;                         // Maximum HR
value to be used for calculation of HR limit values.
                                                                  // HR
value in bpm, 100 - 240 bpm

        POLAR_EXEPHASE  Phase[6];       // One exercise profile includ
es
6 exercise phases
                                                                  // Eac
h
phase should be defined as POLAR_EXEPHASE structure

} POLAR_SSET_EXERCISEPROFILE;

__declspec (dllexport) BOOL CALLBACK fnHRMCom_ResetExerciseProfile
(int, POLAR_SSET_EXERCISEPROFILE*, int);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendExerciseProfile
(POLAR_SSET_GENERAL*, POLAR_SSET_EXERCISEPROFILE*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_ReadExerciseProfile
(POLAR_SSET_GENERAL*, int, POLAR_SSET_EXERCISEPROFILE*);

//      NOTE: Exercise profiles are available only with Polar S810 HR
monitor.
//      Integer value at resetting and reading functions include Exe
Profile number (1 - 7)
```

```
//      If sending was succesfull, function returns TRUE, otherwise
FALSE

□

/////////////////////////
/////////////////////////
//
// MONITOR BITMAP LOGO
//
/////////////////////////
/////////////////////////

//      NOTE: Monitor bitmap logos are available with Polar S610, S710
,
S810, E200 and E600 HR monitors.

__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendBitmap

(POLAR_SSET_GENERAL*, int*);
__declspec (dllexport) BOOL CALLBACK fnHRMCom_ReadBitmap

(POLAR_SSET_GENERAL*, int*);

//      If sending was succesfull, function returns TRUE, otherwise
FALSE

// Example:     Each pixel column in one integer value => int
iBitmapPixelCol[47];
//                      First pixel in the bottom is 2^0, second 2^1,
third 2^2, etc.
//                      If three pixels in bottom are ON =>
iBitmapPixelCol[iColumn] = 7 (1+2+4)
//                      If entire column is ON =>
iBitmapPixelCol[iColumn] = 255 (1+2+4+8+16+32+64+128)
//                      Send to monitor fnHRMCom_SendBitmap
(&iBitmapPixelCol[0]);

□

/////////////////////////
/////////////////////////
//
// SET HR MONITOR TO WATCH MODE
//
/////////////////////////
/////////////////////////
```

```
//      Sets monitor to watch mode, monitor do not accept other
//      messages, until it has been switched back to Connect mode.

__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendMonitorToWatchMode
(POLAR_SSET_GENERAL*);

//      If sending was succesfull, function returns TRUE, otherwise
FALSE

□

/////////////////////////
/////////////////////////
//
// FACTORY DEFAULTS
//
/////////////////////////
/////////////////////////

//      Sets monitor factory defaults, resets all monitor data includi
ng
EEPROM memory. Use very carefully!!!
//      Setting factory defaults is not meant for normal software usag
e,
only for service software products.
//      When settings factory defaults, confirmation of the operation
should be asked always.

__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendFactoryDefaultComman
d
(POLAR_SSET_GENERAL*);

□

/////////////////////////
/////////////////////////
//
// DELETING EXERCISE FILES FROM HR MONITOR
//
/////////////////////////
/////////////////////////

//      Exercise files can be deleted by using the following function
call. Files can be deleted
//      from Polar S610, S710, S810 and E600 HR monitors.
//
//      NOTE: The first version of Polar S610 (DataVersion=1) can't
handle deleting one exercise
```

```
//      file correctly, all exercise files can still be deleted.
//
//      NOTE: The first version of Polar S810 (DataVersion=3) can't
handle deleting all exercise
//      files correctly, one exercise file can still be deleted.
//
//      Check monitor type and data version before sending file delete

message to monitor!
//
//      Give exercise number as int parameter (0 - n), if all files
should be deleted, use int parameter -1

__declspec (dllexport) BOOL CALLBACK fnHRMCom_SendFileDeleteCommand
(POLAR_SSET_GENERAL*, int);

□

////////////////////////
/////////////////////////
//
// INFRARED COMMUNICATION FUNCTIONS
//
/////////////////////////
/////////////////////////

/////////////////////////
//
// MONITOR INFO
//
/////////////////////////

typedef struct
{
        int             iSize;                              //
Structure size for version control

        int             iMonitorInUse;                  // HR monitor
in
use: HRM_S610, HRM_S710, HRM_S810 or HRM_E600
        int             iDataVersion;                   // HR monitor
data version

        int             iTotalFiles;                    // Total count

of all files inside HR monitor
        int             iFreeMemoryInBytes;             // Free memory
inside HR monitor (in bytes)
        int             iTotalMemoryInBytes;    // Total memory inside

HR monitor (in bytes)

        BOOL    bLowBattery;                    // Low battery indicat
or
TRUE / FALSE

} POLAR_SSET_MONITORINFO;

__declspec (dllexport) BOOL CALLBACK fnHRMCom_ReadMonitorInfo
(POLAR_SSET_GENERAL*, POLAR_SSET_MONITORINFO*);

□

//////////////////////////////////////////////////////////////////////
//
///////
//
//      fnHRMCom_ResetIRCommunication
//      -----------------------------
//      Call in the startup of software to reset all the communication

parameters.
//
//      Parameters:
//      int iParam              Reserved in future usage, use 0 (zero)
.
//
//      Return value:
//      TRUE                    - Resetting made succesfully
//      FALSE                   - Resetting was not made because of
communication is already running.
//
//////////////////////////////////////////////////////////////////////
//
///////

__declspec (dllexport) BOOL CALLBACK fnHRMCom_ResetIRCommunication
(int);

□

//////////////////////////////////////////////////////////////////////
//
///////
//
```

```
//      fnHRMCom_StartIRCommunication
//      -----------------------------
//      Call to open communication port and start IR communication.
//
//      Parameters:
//      int iParam                 Parameter for connection settings (whe
n
used multiple params, use OR)
//                                         See "Infrared communication
parameters"
//
//      LPTSTR tcPort   Communication port name, for example "COM1:" o
r
"COM2:"
//                                         Remember to use to use colon :

at the end of port name
//
//      Return value:
//      BOOL bStartOK
//       TRUE                      - Starting of communication made
succesfully
//       FALSE                     - Problems encountered, check the
following possible errors:
//                                                 * Communication has
already been started and it is running
//                                                 * Communication port
already reserved for some other device
//                                                 * Maybe call was made
from 16-bit program. A 32-bit DLL cannot
//                                                   create an additional

thread when that DLL is being called by
//                                                   a 16-bit program.
//
//////////////////////////////////////////////////////////////////////////
//
///////

__declspec (dllexport) BOOL CALLBACK fnHRMCom_StartIRCommunication
(int, LPTSTR);

□

// Infrared communication parameters
#define HRMCOM_PARAM_INTERNALIR                 1       // Use interna
l
IR port (Win95 only)
```

```
#define HRMCOM_PARAM_KEEPCONNECT                2       // Keep monito
r
in Connect mode during connection
#define HRMCOM_PARAM_FILTERHRDATA               4       // Filter
averaged HR data (sampling rate 5 or 15 seconds)


// (not yet implemented)
#define HRMCOM_PARAM_DIRECT_USB             16   // Direct USB port usa
ge
#define HRMCOM_PARAM_VIRTUALCOMPORT         32   // Virtual COM port
usage
#define HRMCOM_PARAM_DUMPFRAMES             64   // Dump frames to
c:\frames.txt or c:\all.txt text files


// Dumping can be used for data error detection.
#define HRMCOM_PARAM_ONLINE                          128   // Online
recording mode (Polar S810 only)

□

///////////////////////////////////////////////////////////////////////
//
///////
//
//       fnHRMCom_EndIRCommunication
//       -----------------------------
//       Call to close communication port and end IR communication.
//
//       Parameters:
//       int iParam                  - Reserved in future usage, use 0
(zero).
//
//       Return value:
//       BOOL bEndOK
//        TRUE                       - Ending of communication made
succesfully
//        FALSE                      - Problems with ending of communicatio
n
//
///////////////////////////////////////////////////////////////////////
//
///////

__declspec (dllexport) BOOL CALLBACK fnHRMCom_EndIRCommunication
(int);
```

□

```
//////////////////////////////////////////////////////////////////////
//
///////
//
//      Communication Texts
//      -------------------
//
//      Communication texts are shown with infrared communication
process. By default
//      English texts for buttons and message texts are defined. If
texts need localization,
//      the following functions can be used to change communication
texts before calling
//      communication functions. The text at the end of the following
defines shows the
//      default text string for each text item.
//
#define         HRMCOM_TEXT_CANCEL                                  0

// Cancel
#define         HRMCOM_TEXT_RETRY                                   1

// Retry
#define         HRMCOM_TEXT_READING                                 2

// Reading...
#define         HRMCOM_TEXT_NOANSWER                        3
// No answer from HR Monitor
#define         HRMCOM_TEXT_ERRORS                                  4

// Errors with Connection
#define         HRMCOM_TEXT_STARTING                        5
// Starting Connection...
#define         HRMCOM_TEXT_TITLE                                   6

// Infrared Connection
#define         HRMCOM_TEXT_WRITING                                 7

// Writing...
//
//      To set each communication text, call function
fnHRMCom_SetComText.
//      For example this call will change internal text for informing
user about
//      not getting any answers from HR monitor within answer time:
//
```

```
//         fnHRMCom_SetComText (HRMCOM_TEXT_NOANSWER, "Ei vastausta
sykemittarilta");
//
__declspec (dllexport) BOOL  CALLBACK   fnHRMCom_SetComText
(int, LPTSTR);
//
//       When HRMCOM.DLL is initialized by starting software calling it
,
all the communication
//       texts are resetted automatically. To reset all communication
texts back to English
//       default texts, the following reset function can be used:
//
__declspec (dllexport) void  CALLBACK   fnHRMCom_ResetComTexts  (void)
;
//
//       NOTE: Title text for Polar UpLink Communication is always "Pol
ar
UpLink".
//
////////////////////////////////////////////////////////////////////////
//
///////

□

////////////////////////////////////////////////////////////////////////
//
///////
//
//       Reading Exercises Data
//       ----------------------
//
//       Reading exercises data from HR monitor using IR connection can

be done by
//       sending and answering to each communication message and also b
y
calling function,
//       which generates communication dialog and handles all message b
y
itself.
//       Communication port has to be opened before reading exercises
from HR monitor.
//       To read all exercises to memory of DLL, use the following
function:
//
__declspec (dllexport) BOOL CALLBACK fnHRMCom_ReadExercisesData
```

```
(HWND, BOOL);
//
//      Parameters:
//              HWND    hOwnerWnd       - Handle to owner window
//              BOOL    bOneWay         - Flag for one way connection
(under construction)
//
//      After all exercises have been read from HR monitor, the basic
information
//      about each exercise can be read by using the following functio
n
fnHRMCom_GetExeFileInfo.
//      Structure POLAR_EXERCISEFILE includes basic information about
the exercise data
//      file requested.
//
typedef struct
{
        int             iSize;                                  //
Structure size for version control

        int             iTime;                                  // Sta
rt
time of exercise in seconds
        int             iDate;                                  // Sta
rt
date of exercise in yyyymmdd
        int             iDuration;                              //
Duration of exercise in seconds
        BOOL    bUSTimeMode;                    // Usage of 12h time
mode in exercise
        int             iSamplingRate;                  // Sampling ra
te
of exercise
        BOOL    bDeleted;                               // Exercise ha
s
been marked to be deleted

        BOOL    bSpeed;                                 // Speed senso
r
data available
        BOOL    bCadence;                               // Cadence
sensor data available
        BOOL    bAltitude;                              // Altitude
sensor data available
        BOOL    bPower;                                 // Power senso
r
data available
```

```
        BOOL    bInterval;                              // Interval da
ta
available

        TCHAR   szName[9];                              // Exercise
set/profile name used in exercise

// Max number of characters is 8 + ending zero

} POLAR_EXERCISEFILE;
//
__declspec (dllexport) BOOL CALLBACK fnHRMCom_GetExeFileInfo
(int, POLAR_EXERCISEFILE*);
//
//      Parameters:
//      int iExercise                          Parameter for specifyi
ng
exercise of which the information will be retrieved
//      POLAR_EXERCISEFILE* pef*        Address to exercise file
information data structure
//
//      Before reading detailed exercise information from HRMCOM.DLL's

memory, each
//      exercise file have to be analyzed by using the following
function:
//
__declspec (dllexport) BOOL CALLBACK    fnHRMCom_AnalyzeFile
(int, int);
//
//      Parameters:
//      int iExercise   Parameter for specifying exercise to be analys
ed
//      int iAction             Parameter for specifying the actions t
o
be doen for analysed HR file
//                                      (when used multiple params, us
e
OR)
//
//                                      HRMCOM_PARAM_FILTERHRDATA
= Filter averaged HR data (sampling rate 5 or 15 seconds)
//

Not available yet!
//
//      After the succesfull analyzing, all the exercise information c
```

```
an
be read
//      by using the functions and defines shown in the following
chapters.
//
//////////////////////////////////////////////////////////////////////
//
///////

□

///////////////////////////////////////////////////////////
//
// HRM DATA OUTPUT FUNCTIONS
//
///////////////////////////////////////////////////////////

__declspec (dllexport) int  CALLBACK    fnHRMCom_GetRecParam
(int);          // returns recording parameters
__declspec (dllexport) BOOL CALLBACK    fnHRMCom_GetRecFlags
(int);          // returns recording flags

__declspec (dllexport) int  CALLBACK    fnHRMCom_GetNbrOfHRMSamples
(void);         // returns nbr. of samples
__declspec (dllexport) int  CALLBACK    fnHRMCom_GetHRMSamples
(int, int);     // returns HR/CC samples

__declspec (dllexport) int  CALLBACK    fnHRMCom_GetNbrOfIntTimes
(void);         // returns number of lap times
__declspec (dllexport) int  CALLBACK    fnHRMCom_GetIntTimeData
(int, int);     // returns lap time data

__declspec (dllexport) BOOL CALLBACK    fnHRMCom_GetNbrOfSwapTimes
(void);         // returns number of HR limit swaps
__declspec (dllexport) int  CALLBACK    fnHRMCom_GetLimitSwapData
(int, int);     // returns limit swap data

///////////////////////////////////////////////////////////
//
// HRM DATA FLAGS
//
//      Get these parameters by using function: fnHRMCom_GetRecFlags
//
///////////////////////////////////////////////////////////

#define         FLAG_CYCLO_DATA                 3

// TRUE, cycling data
```

```
#define         FLAG_3LIMITS_IN_USE                  6

// three HR limits has been used
#define         FLAG_SPEED_DATA                  8

// file has speed data
#define         FLAG_ALT_DATA                    9

// file has altitude data
#define         FLAG_CAD_DATA                    10

// file has cadence data
#define         FLAG_POWER_DATA                  11

// file has power data
#define         FLAG_INTERVAL_DATA               12

// file has interval data
#define         FLAG_LAP_DATA                    13

// file has lap data
#define         FLAG_LIMSWAP_DATA                14

// file has limit swap data
#define         FLAG_POWER_BALANCE               18

// file has LR balance data
#define         FLAG_POWER_INDEX                 19

// file has pedalling index data


////////////////////////////////////////////////////////////
//
// HRM DATA GENERAL RECORDING INFORMATION
//
//      Get these parameters by using function: fnHRMCom_GetRecParam
//
////////////////////////////////////////////////////////////

#define         REC_AM_PM
1                                   // 0 = AM, 1 = PM
#define         REC_MONITOR_TYPE                                5

// HR Monitor Type
#define         REC_EURO_US_UNITS                               8

// 0 = Euro, 1 = US
```

```
#define         REC_START_DATE                                     9

// Exercise start date in yyyymmdd format
#define         REC_START_TIME                                     10

// Exercise start time hh:mm:ss.s/10 in 1/10 of seconds
#define         REC_REC_LENGTH                                     11

// Duration on exercise (in ms)
#define         REC_SAMPLING_RATE                                  12

// Recording rate
#define         REC_UPPER_LIMIT_1                                  13

// 0 - 250 bpm
#define         REC_LOWER_LIMIT_1                                  14

// 0 - 250 bpm
#define         REC_UPPER_LIMIT_2                                  15

// 0 - 250 bpm
#define         REC_LOWER_LIMIT_2                                  16

// 0 - 250 bpm
#define         REC_UPPER_LIMIT_3                                  17

// 0 - 250 bpm
#define         REC_LOWER_LIMIT_3                                  18

// 0 - 250 bpm
#define         REC_ANAEROB_LIMIT                                  19

// 0 - 250 bpm
#define         REC_AEROB_LIMIT                                    20

// 0 - 250 bpm
#define         REC_TIMER_1
21                              // timer 1 in seconds
#define         REC_TIMER_2
22                              // timer 2 in seconds
#define         REC_TIMER_3
23                              // timer 3 in seconds
#define         REC_MAX_HR
25                              // UpperLimit+1 - 250
#define         REC_REST_HR
26                              // 0 - LowerLimit-1
#define         REC_RR_START_DELAY                                 27
```

```
// R-R recording start delay
#define            REC_START_SAMPLE                                29

// 0 - 250 bpm

#define            REC_STOP_TIME                                   30

// hh:mm:ss.s/10 in 1/10 of seconds
#define            REC_STOP_SAMPLE                                 31

// 0 - 250
#define            REC_STOP_SPEED                                  32

// stop speed
#define            REC_STOP_CAD                                    33

// stop cadence
#define            REC_STOP_ALT                                    34

// stop altitude
#define            REC_MIN_HRATE                                   35

// lowest heart rate
#define            REC_AVE_HRATE                                   36

// average heart rate
#define            REC_MAX_HRATE                                   37

// highest heart rate

#define            REC_TRIP_DIST_STOP                              38

// trip distance at stop
#define            REC_TRIP_CLIMB_STOP                             39

// trip climb at stop
#define            REC_TOT_TIME_STOP                               40

// total time at stop
#define            REC_AVG_ALT
41                                 // average altitude
#define            REC_MAX_ALT
42                                 // maximum altitude
#define            REC_AVG_SPEED                                   43

// average speed
#define            REC_MAX_SPEED                                   44
```

```
// maximum speed
#define         REC_ODOM_STOP                                   45

// odometer stop
#define         REC_MIN_SPEED                                   46

// minimum speed

#define         REC_RECOVERY_TIME                               47
#define         REC_RECOVERY_HR                                 48
#define         REC_MAX_POWER                                   78

// Maximum power in watts
#define         REC_AVE_POWER                                   79

// Average power in watts
#define         REC_CALORIES                                    80

// Calory consumption
#define         REC_NBR_OF_LIMITS_IN_USE                83

// Nbr. of HR limits in use


//////////////////////////////////////////////////////////////
//
// HRM DATA SAMPLE TYPES
//
//      Before getting measured values (samples), get the number of
samples by using
//      function fnHRMCom_GetNbrOfHRMSamples. After this operation,
samples can be get
//      by calling function fnHRMCom_GetHRMSamples for example in the
following way:
//
//      iTotal = fnHRMCom_GetNbrOfHRMSamples ();
//
//      for (i = 0; i < iTotal; i++)
//      {
//              iHR[i]    = fnHRMCom_GetHRMSamples (CC_HRATE, i);
//              iSpeed[i] = fnHRMCom_GetHRMSamples (CC_SPEED, i);
//              iCad[i]   = fnHRMCom_GetHRMSamples (CC_CAD, i);
//      }
//
//      Speed and altitude values unit depends of recording parameter
REC_EURO_US_UNITS.
//      To get the correct units, use for example the following call:
//
```

```
//      if (1 == fnHRMCom_GetRecParam (REC_EURO_US_UNITS))
//      {
//              Speed in mph, altitude in feet
//      }
//      else
//      {
//              Speed in km/h, altitude in meters
//      }
//
/////////////////////////////////////////////////////////////

#define         CC_HRATE
1                                   // heart rate values (bpm / msec)
#define         CC_SPEED
2                                   // speed values (10 * km/h / 10 * mph)
#define         CC_CAD
3                                   // cadence values (rpm)
#define         CC_ALT
4                                   // altitude values (m / ft)
#define         CC_POWER
5                                   // power values (Watts)
#define         CC_POWER_BALANCE                                  6

// power LR Balance (left%)
#define         CC_POWER_INDEX                                    7

// power pedalling index (%)


/////////////////////////////////////////////////////////////
//
// LAP TIME DATA INFORMATION
//
//      Before getting lap time data, get the number of laps by using
//      function fnHRMCom_GetNbrOfIntTimes. After this operation, lap
//      information can be get by calling function
fnHRMCom_GetIntTimeData
//      for example in the following way:
//
//      iTotal = fnHRMCom_GetNbrOfIntTimes ();
//
//      for (i = 0; i < iTotal; i++)
//      {
//              iTime  = fnHRMCom_GetIntTimeData (i, INT_INT_TIME);
//              iHR    = fnHRMCom_GetIntTimeData (i, INT_SAMPLE);
//              iSpeed = fnHRMCom_GetIntTimeData (i, INT_SPEED);
//      }
//
```

```
//////////////////////////////////////////////////////////////

#define          INT_INT_TIME                                         601

// Lap time in 1/10 seconds
#define          INT_LAP_INTRVAL                                      603

// Lap type: 0 = normal lap, 1 = interval
#define          INT_LAP_DISTANCE                                     604

// Lap distance in meters / yards
#define          INT_SAMPLE
607                                // Momentary HR, 0 - 250 bpm
#define          INT_MIN_SAMPLE                                       608

// Lap's min HR, 0 - 250 bpm
#define          INT_AVE_SAMPLE                                       609

// Lap's avg HR, 0 - 250 bpm
#define          INT_MAX_SAMPLE                                       610

// Lap's max HR, 0 - 250 bpm
#define          INT_SPEED
611                                // Momentary speed, 10 * km/h or mph
#define          INT_AVG_SPEED                                        612

// Average speed, 10 * km/h or mph
#define          INT_CADENCE
613                                // Momentary cadence, 0 - 180 rpm
#define          INT_AVG_CADENCE                                      614

// Average cadence, 0 - 180 rpm
#define          INT_ALTITUDE                                         615

// Momentary altitude, (-1000 - 2047) * 10 m / ft
#define          INT_AVG_ALTITUDE                                     616

// Average altitude, (-1000 - 2047) * 10 m / ft
#define          INT_POWER
617                                // Momentary power, 0 - 2000 Watts
#define          INT_MAX_POWER                                        618

// Maximum power, 0 - 2000 Watts
#define          INT_AVE_POWER                                        619

// Average power, 0 - 2000 Watts
#define          INT_TEMP
621                                // Momentary temperature, 10 * -100 -
```

```
+100 'C or 'F (only from S710)
#define         INT_DIST_REC                                      624

// Distance recovery, 10 * km or miles
#define         INT_RECOVERY                                      625

// Recovery calculation, 0 = No recovery, 1 = Time Recovery, 2 = +HR
Recovery, 3 = -HR Recovery, 4 = Distance Recovery
#define         INT_HR_REC
626                         // HR recovery value, 0 - 3599 seconds
#define         INT_TIME_REC                                      627

// Time recovery value, 0 - 240 bpm
#define         INT_LAP_ASCENT                                    636

// Lap ascent, trip up,  m / feet

//////////////////////////////////////////////////////////////////////////
//
///////
//
// HR LIMIT SWAPS, indexes for 'fnHRMCom_GetLimitSwapData'
//
//////////////////////////////////////////////////////////////////////////
//
///////

#define         LIM_SWAP_TIME                                     900

// HR limit swap time
#define         LIM_SWAP_CODE                                     901

// HR limit swap code

/////////////////////////
/////////////////////////
//
// ONLINE RECORDING
//
/////////////////////////
/////////////////////////

//      Online recording is available only with Polar S810 HR monitor.

To start online recording,
//      function fnHRMCom_StartIRCommunication have to be called with
parameter HRMCOM_PARAM_ONLINE.
//      Function 'fnHRMCom_GetOnlineData' returns online data samples
```

```
received from the S810
//      HR monitor or ONLINE_BUFF_EMPTY if there aren't any new online

samples in the buffer.
//
//              iData = fnHRMCom_GetOnlineData (iParam);
//
//      Parameter 'iParam' is 32-bit integer and it is reserved for
future use and it should
//      be 0 (zero) now. Return value is 32-bit integer and it is R-R
value in milliseconds
//      or ONLINE_BUFF_EMPTY if there aren't any new samples in buffer
.
//
////////////////////////////////////////////////////////////

#define         ONLINE_BUFF_EMPTY                                  -1

// online buffer is empty

__declspec (dllexport) int  CALLBACK    fnHRMCom_GetOnlineData
(int);          // return online data samples
```